THE UNIVERSITY OF WAIKATO
Department of Computer Science

COMP204-24B — Practical Networking & Cyber Security

A3 - TFTP

---

1 Introduction

3.1 Skeleton code

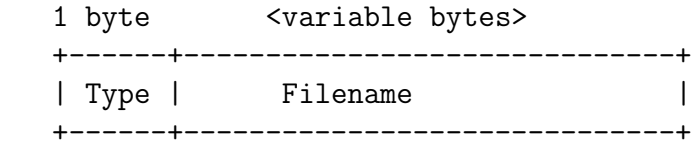
We provide the basic skeleton of a TftpServer implementation to get you started. You do not have to use it. We have left comments that describe roughly the process you should use. Briefly, the simplest implementation strategy is where the server responds to each TftpClient request using a different thread. You need to implement a retransmission method, in case a data or acknowledgment packet is lost, and you should use the DatagramSocket::setSoTimeout method for this.

3.2 Packet Formats

Every packet begins with a 1 byte type field. The defined types are:

- RRQ: 1
- DATA: 2
- ACK: 3
- ERROR: 4

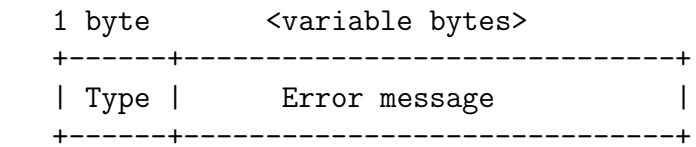
To start with, implement the "Read Request" (RRQ) exchange in both the client and the server. Have the server listen on a port it can bind to (not 69). The RRQ packet contains a String of the requested filename after the type field:



Format of RRQ packet.

To obtain a byte array from a String, use the getBytes() method on the String. To generate a String from a byte array, use the appropriate String constructor. Use the DatagramPacket::getLength field to determine the size of the packet, and thus be able to compute the size of the filename, in bytes.

If the file does not exist on the server, send an error packet back. The ERROR packet contains a String that could explain why the request could not be fulfilled, such as because the file does not exist. Print the error message in the client.



Format of ERROR packet.

If the file does exist, create a new thread in the server which will send the DATA back using a new DatagramSocket. Send the first block using block number #1. Put up to 512 bytes of data in each packet: a full size DatagramPacket, including the type and block number, contains 514 bytes of data. Send the DatagramPackets to the IP and port number the client made the request from.

```
1 byte 1 byte        <variable bytes>
+------+-------+-------------------------+
| Type | Block |      Data               |
+------+-------+-------------------------+
Format of DATA packet.
```

The client sends an ACK packet when it receives a DATA packet it expects: the block number has to be either:

1. the block it is expecting, or
2. the block it received last time.

The second condition occurs if the ACK is lost in the network being carried back to the server. The block number in the ACK packet is the in-order packet your client last received.

```
1 byte 1 byte
+------+-------+
| Type | Block |
+------+-------+
Format of ACK packet.
```

Note that the block number will wrap to zero as the maximum value that can go into a byte is 255. To test your implementation, transfer a file at least 131,072 bytes in size. Verify that the received file is correct using the md5sum command to compare the checksums of the sent and received files.

In the server, use setSoTimeout on the DatagramSocket object to set a 1 second timeout before calling the recv method to wait for an ACK. The DatagramSocket will throw an SocketTimeoutException if it does not receive an ACK in a second.

3.3 Interoperability

You should test your client and server implementation with that of a classmate. Ensure they behave correctly. If they do not, work with your classmate to determine whose implementation (or both) has a problem according to the specification provided.